AMENDMENT TO THE SPECIFICATION

Please amend the specification by marked up replacement paragraph(s) as follows.

Page 1, lines 8-11:

The present application claims the benefit of U.S. Provisional Patent Application Serial Number ~~________~~ 60/198,258, entitled "Persistent Agents" filed on December 17, 1999 by Rafiul Ahad ~~(attorney docket number 50277-404)~~, the contents of which are hereby incorporated by reference.

Page 1, lines 16-21:

Typically, most objects created by a process are destroyed when that process terminates. A persistent object, however, is an object whose lifetime can exceed the lifetime of the process that creates or accesses the object. Object persistence is therefore a useful property for those objects instantiated in one process that carry information needed by other processes. For example, it may be desirable for one ~~Java~~ JAVA$^{\text{TM}}$ application to access the objects previously created by another Java application.

Page 3, lines 14-24:

Attempts have been made to provide the implementation of persistence in a "persistent object" base class. Thus, the programmer designing the classes for persistent objects must derive these class from the persistent object base class. In single-inheritance languages such as ~~Java~~ JAVA$^{\text{TM}}$, however, this attempt compels the programmer to derive almost every class from the persistent object base class, which adds persistence to the interface of all of such classes, even if unneeded for particular classes, just to permit the use of polymorphism. Another attempt to reduce the programming overhead is by special pre-processing or post-processing of nonpersistent classes to make the instances of the classes persistent. However, such pre-processing and post-processing greatly complicates the maintenance of the software and creates further opportunities for error when the pre-processing and post-processing are not consistently applied.

Page 9, lines 8-16:

Computer system 100 may be coupled via bus 102 to a display 112, such as a cathode ray tube (CRT), for displaying information to a computer user. An input device 114, including alphanumeric and other keys, is coupled to bus 102 for communicating information and command selections to processor 104. Another type of user input device is cursor control 116, such as a mouse, a trackball, or cursor direction keys for communicating direction information and command selections to processor 104 and for controlling cursor movement on display 112. This input device typically has two degrees of freedom in two axes, a first axis (e.g., x) and a second axis (e.g., y), that allows the device to specify positions in a plane. Another input device can be a microphone 117.

Page 12, lines 4-9:

In accordance with one embodiment of the present invention, the following persistent object model is adopted within a ~~Java~~ JAVA™ programming language environment. Although the following description refers to the ~~Java~~ JAVA™ programming language environment, it is to be understood that the concepts of the present invention are not necessarily limited to the Java programming language environment, but, on the contrary, may be applied in other object-orient programming language environments as well.

Page 12, lines 10-21:

Under such a persistent object model, any object can be stored persistently in a persistent object store. A persistent object store is a computer-readable medium providing a long-term storage, such as a file system or database, capable of storing objects beyond the lifetime of any process that created or accessed the objects. In some implementations, the persistent object store can be chosen to provide data synchronization for a mobile user using a laptop, such as by storing persistent objects in updateable snapshots that are replicated to a host database system when the mobile user dials up the host database system. Updateable snapshots are described in more detail in a mobile user environment in the co-pending, commonly assigned U.S. Patent Application Serial No. 09/322,152 entitled "Data Replication for Front Office Automation" filed on May 28, 1999 by Souder et al., now U.S. Patent No. 6,532,479, issued March 11, 2003, whose contents are incorporated by reference in their entirety.

Page 12, line 22–page 13, line 2:

For appropriate implementations of the persistent object store, especially for databases, all operations against a persistent object store are done within a "session," which represents a connection to the persistent object store and provides appropriate transaction semantics (e.g. commit and rollback) for those operations. In one example, a session is created by calling a "connect" method of the persistent object store interface and is thus represented by a ~~Java~~ JAVA™ object that implements a "persistent object store session" (POSSession) interface.

Page 13, line 23–page 14, line 8:

In one implementation, the fields of the ~~Java~~ JAVA™ object can be identified based on the fully-qualified ~~Java~~ JAVA™ class name by using the Java Reflection API. Thus, the values of the various fields are read from the object using the Java Reflection API and stored in the persistent object store, for example, by appropriate SQL commands. If any of the fields refer to an object of a ~~Java~~ JAVA™ reference type that is not user-defined (e.g. String or int), the persistent agent makes a copy of that object and stores it in the persistent object store. If any of the fields refers to an object of a user-defined class, on the other hand, an already instantiated persistent agent for that user-defined class may be used to retrieve the reference to the object which is then stored in the persistent object store. However, if there is no such persistent agent instantiated for the user-defined class, a default persistent agent for the user-defined class can be automatically created with default properties or simply ignored. This behavior can be specified by setting a property for the persistent agent.

Page 15, line 14–page 16, line 5:

Invoking a `createPersistentAgent()` method creates the necessary data structures in the persistent object store, such as tables in a database, and returns an instantiated persistent agent for a given ~~Java~~ JAVA™ class. The `javaClassRef` parameter is a fully-qualified name of the class, so that the Java Reflection API can be used to identify and access the fields of instances of the class. The `props` parameter is for specifying the properties of the persistent agent. For example, the following properties may be supported:

Page 20, lines 12-18:

In addition, a `getFields()` method is provided to returning the names of the fields defined on the persistent agent and thus the corresponding object. In a ~~Java~~ JAVA™ implementation of this interface, all declared and inherited, private, protected, package, and public fields must be returned that are not declared to be transient and that are not inherited from classes in the Java.lang package. Finally, a `getType()` method is supplied for providing the fully-qualified name of the type or class of a particular field.